一种混合式僵尸主机检测算法的设计与实现

周振吉,吴礼发,梁其川,李华波
(解放军理工大学 指挥自动化学院,江苏 南京 210007)

摘 要:为了进一步提高检测的精确性,在研究僵尸主机的行为特点以及僵尸网络命令与控制信道的特性后,提出了一种基于终端系统行为和网络行为的混合式僵尸主机检测算法,并对现有的僵尸网络行为稳定性衡量方法进行了改进。在此基础上,设计实现了一个僵尸主机检测原型系统——BotScout。评估结果表明了算法的有效性。

关键词:系统行为;网络行为;僵尸网络;僵尸主机

中图分类号:TP393 **文献标识码**:A **文章编号**:1674-7720(2010)16-0057-04

Design and implementation of a hybrid detection algorithm of zombie

ZHOU Zhen Ji, WU Li Fa, LIANG Qi Chuan, LI Hua Bo
(ICA, PLAUST, Nanjing 210007, China)

Abstract: To improve the accuracy of detection, this paper analyzes the properties of Botnet's command and control channel as well as the bot's behavior characteristics, and proposes a detection algorithm which combines system behavior and network behavior to detect zombies, and improves the measure method of stability of Botnet's control flows. A zombie detection prototype system called BotScout is implemented and experiment results show that the algorithm is effective to detect all types of zombies.

Key words: system behavior; network behavior; Botnet; zombie

近年来,僵尸网络已经成为互联网稳定和安全的最大威胁,国内外安全界对此给予了高度关注。僵尸网络(Botnet)是僵尸主人(BotMaster)通过命令与控制信道(C&C)控制的具有协同性的恶意计算机群,其中被控制的计算机称为僵尸主机(Zombie),僵尸主人用来控制僵尸主机的计算机程序称为僵尸程序(Bot)。通过这个计算机群,僵尸主人可以实现覆盖面更广、强度更高、更难被检测的恶意行为,例如分布式拒绝攻击(DDoS)、发送垃圾邮件(Spam)、窃取敏感信息等。

僵尸网络具有以下3个主要特性[1]:(1)"恶意性",即僵尸主机主要开展DDoS、Spam、下载延伸其他恶意代码等一系列恶意行为;(2)"可控性",即僵尸主机通过C&C信道接收僵尸主人的命令后执行相应的恶意行为;(3)"协同性",即同一个僵尸网络中的僵尸主机联合同步执行僵尸主人的命令。

目前,针对最早出现的IRC僵尸网络的检测技术取得了比较理想的效果[2]。但是近几年来,P2P和HTTP类型的僵尸网络日益盛行,而P2P协议多样性和HTTP协议普及性的特点,使得传统的僵尸网络检测技术面临巨大的挑战。本文从终端层面出发,提出了一种基于主机系统行为和网络行为的混合式检测算法,该算法可以有效检测出IRC、P2P和HTTP类型的僵尸主机。

1 相关研究

目前,大多数僵尸网络检测技术主要是从僵尸网络的网络流量特性入手。

GU G等人[1,3]提出了基于网络特征的僵尸网络检测方法,实现了原型系统BotSniffer和BotMiner。将被监视的网络内部每台主机的通信行为和网络恶意活动进行分类,找出具有相似或相关通信和网络恶意行为的主机。如果被监控的内部网络有较多的主机而其中只有少数几台(即使一台)感染了僵尸程序,这种方法就失去了检测的意义。

KANG J等人[4]提出了一种多图累加和(Multi-chart CUSUM)检测P2P僵尸网络的方法。认为主机产生的网络流量是一个复杂的随机模型,发生任何异常都会给这个模型带来明显的变化。而一台主机被感染P2P僵尸程序后,会表现出明显的异常网络流。作者用多图累加和模型描述网络异常,并采用动态阈值自适应技术提高检

测的精确性。

NOH S 等人[5]提出了一种使用多定相流模型(Multi-Phased Flow Model)检测 P2P 僵尸网络的方法,对 P2P 僵尸程序与其他节点通信的多种流量特征进行了分析,然后用相似度进行判断和检测。该方法充分利用了 P2P 僵尸网络的网络流异常实施检测,但是对其他类型僵尸网络检测效果不好,专用性太强。

WANG B 等人[6]利用 P2P 僵尸网络控制流的稳定性对僵尸主机进行检测。他们从 P2P 僵尸网络的控制流中选取一个变量,该变量在整个时间段的值都稳定在某一个水平,而且上下波动的幅度不大。这种方法的误报率和漏报率比较高。

除了僵尸网络的网络行为特性外,LIU L 等人[7]从僵尸程序执行特征出发,提出了一个检测原型系统 BotTracer。由于没有考虑僵尸程序的网络控制流特性,误报率比较大。而 HOLZ T 等人[8]采用蜜网蜜罐技术,但是这类方法依赖于蜜网和蜜罐的分布,无法有效地检测出全部活跃的僵尸网络。

2 算法设计

通过分析大量僵尸程序样本与综合现有文献[9-11]得知:僵尸程序在系统行为和网络行为方面与正常程序都有比较明显的区别,而这些区别可以作为僵尸程序检测的依据。

僵尸程序为了自启动和执行恶意行为(包括键盘记录、密码盗取、网络嗅探、私密后门安装、间谍软件和 rootkit 等),必须调用特定的系统函数。例如网络下载命令会从终端向外发起连接,向目标请求数据,并且在本地终端上面创建新文件。所有这些动作(网络连接、发送、接收、文件创建)都通过调用系统函数来实现,而诸如下载地址这样的控制信息是从网络中接收到的。正常的网络应用程序虽然也接收网络数据,但是一般不会从接收到的数据中提取参数来调用这些特殊的系统函数[12-13]。因此通过监控特定函数调用的参数来源,就可以判定出僵尸主机在终端系统上的恶意行为。

僵尸主机的网络流量按照用途可以分成两类:(1)行为流,指与僵尸主机恶意网络行为相关的流量(如 DDoS、Spam、扫描(Scan)等产生的流量);(2)控制流,指与获得僵尸主人命令、维持 C&C 信道等相关的流量。通过研究发现,与 C&C 信道建立连接后,僵尸主机为了保持连接的活跃,一般会周期性地发送特定的报文,这中间没有用户的干预,因而控制流表现出一定的稳定性,而正常的网络应用程序运行时,由于存在用户干预,一般不会表现出这样的稳定性,因此控制流的稳定性可以作为僵尸网络的一个重要判断依据。

本文在综合考虑僵尸主机的系统行为和网络行为的基础上,提出了一种混合式僵尸主机检测算法。僵尸主机检测算法监控有网络通信行为的进程,并判断其是否为僵尸进程,只要存在僵尸进程,则该终端为僵尸主机。僵尸进程检测算法主要结合程序自启动、控制流的稳定性、系统恶意行为和网络恶意行为 4 个指标来判定。僵尸进程检测算法如下所示:

```
输入:有网络通信行为的进程
输出:是否为僵尸进程
过程:
if 系统恶意行为||网络恶意行为
    if 自启动 && 控制流稳定
        报警;
        exit;
    else
        可能是下载者、木马等其他恶意代码;
    end if
else
    什么都不做;
end if
```

2.1 系统行为监控算法设计

根据上面的分析,本节给出系统行为监控算法,算法中 BotBehavior 变量记录恶意行为数,X_0 是根据实践预定义的阈值。恶意行为累计达到一定的阈值时,则标记为系统恶意特征。系统行为监控算法如下所示:

```
输入:Windows 操作系统某一进程和该进程接收到的网络数据
输出:报警信息(也可无报警)
过程:
BotBehavior=0;      //初始化时,僵尸行为数为 0;
SelectAPI 1=AutoAPISet;//APISet 是一些自启动行为必须调用的系统函数
SelectAPI 2=MalAPISet;//APISet 是一些恶意行为必须调用的系统函数
X=X0;               //给判定阈值赋初值
while 进程为终结
    捕获该进程的系统调用 call;
    if call 属于 SelectAPI1
        报警自启动
    else if call 属于 SelectAPI2
        if call 的参数是接收自网络的数据
            BotBehavior ++;
        end if
    end if
    if BotBehavior>=X
        报警系统恶意行为;
        exit;
    end if
end while
```

2.2 网络行为监控算法设计

控制流的稳定性是判定僵尸网络的一个重要依据。基于参考文献[6]中提出的稳定性思想,本节设计了一

个改进的流量稳定性衡量方法。

2.2.1 稳定性的衡量

在一个时间段 E 内，所有的 m 个流(flow)组成一个流量 C。定义流量 $C=\{\text{flow}_j\}, j=1,2,\cdots,m$，其中 flow_j 表示单个流。为了检测稳定性，需要选择一个合适的随机变量。在实验中，选择最简单的分组平均字节数 abp(the average number of bytes per packet)作为随机变量。分组平均字节数的计算：一个 flow_j 的总字节数除以总分组数。对于一个流量 C，可以计算出 abp 在时间段 E 中任何子集上的离散样本分布。假设对于相邻的时间间隔 T_1 和 T_2，abp 的值分别为 P_1 和 P_2，定义两者之间的距离函数 $S(P_1,P_2)$。

距离函数衡量了两个不同时间间隔内 abp 的变化情况。在稳定性检测算法中，要通过 S 来检测通信流量是否稳定。因此，距离函数 S 的选取至关重要。距离的衡量要足够敏感，以至可以区分同一个数据集中两个样本的差异；但又不能过于敏感，以免无法描述稳定性，因为不同数据集中的样本间的距离比较大。通过实验比较，本文选取距离函数 $S(x,y)=\sqrt{|x-y|}$。结果表明，该距离既能有效区分流量的变化，又不损害稳定性的理解。

2.2.2 稳定性检测算法

本节给出一个在时间段 E 中通信流量 abp 分布的稳定性检测算法。在时间段 E 的流量 C 中使用了两个滑动窗口，一个为基准窗口 W_b，另一个为检测窗口 W_d，两个窗口具有相同的大小。滑动单元用 T_u 表示，而且两个窗口的大小是滑动单元的整数倍。P_b 和 P_d 的含义也和上面提出的距离检测算法相同。在这个算法中引入参数 δ，表示两个 abp 分布的相似度，δ 越小，它们越相似，反之亦然。在检测过程中，δ 也代表了检测标准的严格程度，δ 越小，系统的检测水平越高。δ 根据前面距离检测的结果选定，如果 $S(P_b,P_d)>\delta$，即表示通信流中出现了异常，不具有稳定性。如果当前的 $S(P_b,P_d)\leq\delta$，向前滑动检测窗口 W_d 一步且维持基准窗口 W_b 的位置不变。一旦距离 $S(P_b,P_d)>\delta$，即给出一个流改变的报告，然后把基准窗口移动到流改变的时间点，检测窗口仍然比基准窗口晚一个滑动单元。稳定性检测算法的输入是一个通信流量 C，输出为 C 中流改变的数量。稳定性检测算法如下所示：

输入：通信流量 C

输出：C 中流改变的数量 CN

$(N_w,T_u,\delta)=(N_{w0},T_{u0},\delta_0)$； //赋初值

$CN=0$ //CN 表示 ChangeNumber，即发生流异常的次数

CTime=通信 C 的开始时间；

tag：W_b=从时间 CTime 开始的 N_w 个 T_u；

W_d=比 W_b 晚一个 T_u 的 N_w 个 T_u；

计算 W_b 的 P_b；

计算 W_d 的 P_d；

while 不到通信流 C 的最后

计算 $S(P_b,P_d)$；

if $S(P_b,P_d)>\delta$

CN++；

CTime=当前时间 //即把流改变时间赋给 CTime；

goto tag；

end if

$P_b=P_d$；

滑动 W_dT_u 个单元；

计算 W_d 的 P_d；

end while

output：CN；

3 算法评估

以提出的混合式僵尸主机检测算法为基础，设计并实现了一个检测原型系统——BotScout。BotScout 运行在 Windows 操作系统上，主要由函数调用监控模块、系统行为监控模块、污染传播监控模块和网络行为监控模块组成，总体架构如图 1 所示。

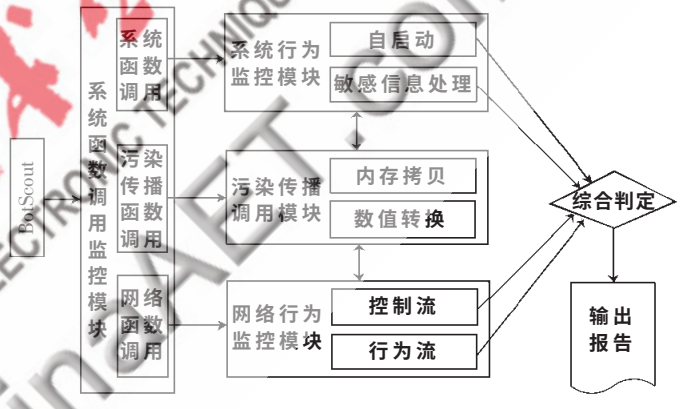

图 1 僵尸主机检测系统 BotScout 总体架构

系统函数对监控模块的调用基于 Microsoft Detours 2.1 Express 设计，根据监控的功能，系统把拦截的 Win32API 函数分为系统行为函数、污染传播函数和网络行为函数三大类。

(1)系统行为监控模块。僵尸程序经常把自己注入到其他进程中以躲避杀毒软件和防火墙的拦截，因此如果某个进程注入代码到其他的进程，则监控被注入进程的行为。僵尸程序为了实现自启动必须修改特定的注册表项或者一些特定文件，而窃取系统的敏感信息也是通过访问特定的文件、注册表、键盘消息实现，监控这些函数的调用就可以发现此类恶意行为。

(2)污染传播监控模块。为检测僵尸程序“可控性”，算法对接收到的网络数据进行监控。网络数据监控发生在网络接收时，这些接收到的数据称为脏数据，算法持续跟踪脏数据：当脏数据写入新的内存空间时，跟踪写入过程，把新的内存区的数据标识为脏数据，并将其加入到脏数据链表中。监控污染的传播相当重要，否则会发生漏报。本模块主要监控内存拷贝函数(如 memcpy)，这也是污染传播的主要途径。其实，还有一些函数也会

起到污染传播的作用，例如把某个内存内容转化为数值的函数（如 atoi）、大小端转化函数（如 htonl、htons），还有一些加密、解密、压缩、解压缩的函数等；并且许多内存拷贝函数是 inline 或者静态链接在文件中的，在原型系统 BotScout 中暂不考虑这些函数，这也是下一步重点工作之一。

(3)网络行为监控模块。如何有效地分离控制流和行为流是这个模块的一个难点。僵尸程序启动后自动与 C&C 信道建立连接，因此标记起始建立的网络连接为控制流，并在整个进程检测周期中对相同协议的网络连接进行统计，分析其稳定性；僵尸主机本质上是控制流驱动行为流，因此如果一个新的网络连接建立过程中地址绑定参数使用了控制流接收到的数据，则标记这个新连接为行为流，并对内容进行分析，检查是否符合 DDoS、Spam、Scan 等恶意行为特征。

3.1 测试环境

由于僵尸网络范围非常广，在实际应用中将整个僵尸网络中的所有节点检测、追踪出来是不可能的，所以僵尸网络检测一般是检测出网络中的部分节点。本文的试验和检测环境为安装有检测系统的局域网，检测对象为局域网内部的主机节点。

BotScout 运行在单台终端系统上，系统测试拓扑如图 2 所示。测试主机通过交换机连接在一起，经过边界路由、防火墙与互联网通信，所有流量控制在此测试环境内，不会对其他主机产生影响。

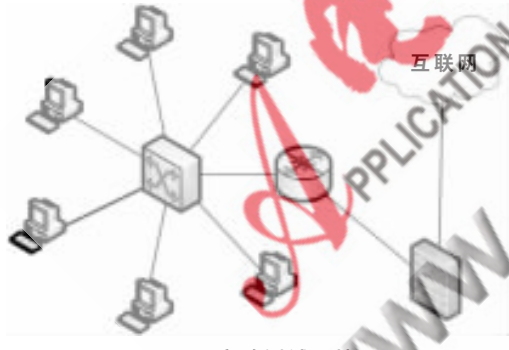

图 2 实验测试环境

共有 6 台测试机，分别对应 6 个样本，每台运行 120 min。这 6 台测试机的运行环境为 Windows XP SP3，配置为 2CPU Intel(R) Core(R) 2.0 GHz、2 GB 内存，100 MB 网卡。

3.2 测试结果

本实验选择了两类样本程序：(1)僵尸程序样本。实验选择了 SDBot、AgoBot 和 SpamThru 三种不同控制协议的僵尸程序；(2)正常网络应用程序样本。实验选择了经典的 IRC 聊天工具 mIRC，热门的网页浏览器 Internet Explorer 和流行的 P2P 下载软件 eMule。对实验数据进行横向测试，结果如表 1 所示，僵尸程序样本都能在较短的时间内被检测出来，而正常网络应用程序在整个运行周期内都没有报警。

表 1 实验检测结果

	mIRC	IE	Emule	AgoBot	BoBax	SpamThru
协议类型	IRC	HTTP	P2P	IRC	HTTP	P2P
总运行时间/min	120	120	120	120	120	120
报警时间/min	无	无	无	8.9	3.5	7

在充分研究僵尸主机的行为特点以及僵尸网络命令与控制信道的特性后，提出了一种综合系统行为与网络行为的混合式僵尸主机检测算法，并对现有的僵尸网络行为稳定性衡量方法进行了改进。基于该算法设计实现了一个僵尸主机检测原型系统 BotScout，并对这个系统的性能进行了测试。实验结果验证了算法的可行性、有效性和准确性。

参考文献

[1] GU G F, PERDISCI R, ZHANG J J, et al. BotMiner: clustering analysis of network traffic for protocol-and structure-independent botnet detection[C]. In Proc. of USENIX Security'08, July 2008.

[2] 王威，方滨兴，崔翔. 基于终端行为特征的 IRC 僵尸网络检测[J]. 计算机学报，2009，32(10)：1980-1988.

[3] GU G, ZHANG J, LEE W. BotSniffer: detecting Botnet command and control channels in network traffic[C]. In Proc. of the 15th Annual Network and Distributed System Security Symposium(NDSS'08), 2008.

[4] KANG J, ZHANG J, LI Q, et al. Detecting new P2P Botnet with multi-chart CUSUM[C]. In 2009 International Conference on Networks Security, Wirelsss Communications and Trusted Computing. 2009.

[5] NOH S, OH J, LEE J, et al. Detecting P2P Botnets using a multi-phased flow model[C]. In Third International Conference on Digital Society. 2009.

[6] WANG B, LI Z, TU H, et al. Measuring Peer-to-Peer Botnets using control flow stability[C]. In 2009 International Conference on Availability, Reliability and Security. 2009.

[7] LIU L, CHEN S, YAN G, et al. BotTracer: execution-based bot-like malware detection[C]. In 11th International Conference on Information Security(ISC 2008). 2008.

[8] HOLZ T, STEINER M, DAHL F, et al. Mesaurments and mitigation of Peer-to-Peer-based Botnets: a case study of storm worm[C]. In Proc. of USENIX LEET'08, April 2008.

[9] NAZARIO J. Botnet tracking: tools, techniques, and lessons learned[C]. In Black Hat, 2007.

[10] ZHUGE J W, HAN X H, ZHOU Y L, et al. Research and development of Botnets[J]. Journal of Software, 2008, 19(1): 152-165.

[11] GRIZZARD J B, SHARMA V, NUNNERY C. Peer-to-Peer botnets: overview and case study[C]. In Proc. of the 1st Workshop on Hot Topics in Understanding Botnets(HotBots 2007). Boston, 2007.

[12] YIN H, SONG D, EGELE M, et al. Panorama: capturing system-wide information flow malware detection and analysis[C]. In ACM Conference on Computer and Communication Security (CCS), 2007.

[13] STINSON E, MITCHELL J C. Characterizing bots' remote control behavior[C]. In Lecture Notes in Computer Science, Volume 4579. Springer Berlin/Heidelberg, 2007.

(收稿日期:2010-03-24)

作者简介:

周振吉,男,1985年生,硕士研究生,主要研究方向:网络与信息系统安全。